JASPERREPORTS - CREATING CHARTS

 Copyright © tutorialspoint.com

Earlier people had to rely on scriptlets to gather the chart data and render the chart using an image element in the report template. JasperReports makes it simple now, as it has a built-in support for charts using the new chart component.

Using the new chart component, user need to apply only the visual settings and define expressions that will help build the chart dataset. JasperReports uses JFreeChart as the underlying charting library. When configuring a new chart component, following three components are involved:

- The overall chart component.
- The chart dataset (which groups chart data-related settings).
- The chart plot (which groups visual settings related to the way the chart items are rendered).

JasperReports currently supports the following types of charts: Pie, Pie 3D, Bar, Bar 3D, XY Bar, Stacked Bar, Stacked Bar 3D, Line, XY Line, Area, XY Area, Stacked Area, Scatter, Bubble, Time Series, High-Low-Open-Close, Candlestick, Multiple Axis, Meter, Thermometer and Gantt.

Chart Properties

Charts are normal report elements, so they share some of their properties with all the other report elements. There is a JRXML element called **<chart>**, used to create each type of chart. This element groups special chart-specific settings that apply to all types of charts.

Chart Sub-Elements

The sub-elements of <chart> element are:

- **<reportElement>**: These are displayable objects like static texts, text fields, images, lines, and rectangles that you put in your report template sections
- **<Box>**: This element is used to surround charts by a border that's customizable on each side.
- **<chartTitle>**: This element is used to place the title of the chart. The *position* attribute decides the title position of the chart in the report. This element has attributes - **Position** (Values could be *Top*, *Bottom*, *Left*, *Right*. Deafult value is *Top*), **color**. <chartTitle> has *font* and *titleExpression* as subelements.
- **<chartSubtitle>**: This element is used to place the subtitle of the chart. This element has attribute - **color**. <chartSubtitle> has *font* and *subtitleExpression* as subelements.
- **<chartLegend>**: The element can control the font-related properties as well as the text color and the background color of the chart legend using this element. This element has attributes - **textColor**, **backgroundColor**
- **<anchorNameExpression>**: This element creates the target for the anchor.
- **<hyperlinkReferenceExpression>**: This element contains a report expression indicating the name of the external resource (usually a URL).
- **<hyperlinkAnchorExpression>**: Hyperlink points to an anchor in an external resource.
- **<hyperlinkPageExpression>**: Hyperlink points to a page in the current report.

- **<hyperlinkTooltipExpression>**: This element controls the ToolTip of hyperlink. The type of the expression should be *java.lang.String*.
- **<hyperlinkParameter>**: This element when present generates a final hyperlink depending on the parameter values.

Chart attributes

Attributes in the <chart> element available for all chart types are:

- **isShowLegend**: This attribute is used to determine if a chart legend will be displayed on the report. Values could be *true*, *false*. Default value is *true*
- **evaluationTime**: Determines when the chart's expression will be evaluated. Values could be *Now*, *Report*, *Page*, *Column*, *Group*, *Band*. Default value is *Now*.
- **evaluationGroup**: This attribute determines the name of the group to be used to evaluate the chart's expressions. The value for this attribute must match the name of the group we would like to use as the chart's evaluation group.
- **hyperlinkType**: This attribute can hold any text value. Default value is *None*. This means neither the text fields nor the images represent hyperlinks, even if the special hyperlink expressions are present.
- **hyperlinkTarget**: This attribute help customize the behavior of the specified link when it is clicked in the viewer. Values could be *Self*, *Blank*. Default value is *Self*
- **bookmarkLevel**: This attribute when set to a positive integer, generate bookmarks in reports exported to PDF. Deafult value is *0*.
- **customizerClass**: This is the name of a class (optional) that can be used to customize the chart. The value for this element must be a String containing the name of a customizer class.

Chart customization

As mentioned above JasperReports uses *JFreeChart* as the underlying charting library. *JFreeChart* contains features that are directly not supported by JasperReports. We can take advantage of these features by supplying a customizer class via the *customizerClass* attribute in <chart> element. A customizer class is nothing but an implementation of the *net.sf.jasperreports.engine.JRChartCustomizer* interface. The easiest way to implement this interface is by extending the *net.sf.jasperreports.engine.JRAbstractChartCustomizer* class and thus having access to parameters, fields, and variables, for more flexible chart customization based on report data.

Chart Datasets

One of the common properties across all chart types is **<dataset>** element. Chart datasets help mapping report data and retrieving chart data at runtime. Each chart type contains different sub-elements to define a chart's expressions that define the data used to generate the chart. All of these sub-elements contain a <dataset> element that defines when the chart's expressions are evaluated and reset.

Several types of chart datasets are available in JasperReports because each type of chart works with certain datasets: Pie, Category, XY, Time Series, Time Period, XYZ, and High-Low. Each of these dataset types implements *net.sf.jasperreports.engine.JRChartDataset* interface that define chart datasets. All chart datasets initialize and increment in the same way, and differ only in the type of data or data series they map.

Dataset Properties

Table below summarizes the attributes of the element <dataset>

Attribute	Description	Values
resetType	This attribute determines when the value of the chart expression is to be reset.	None, Report, Page, Column, Group. Default value is **Report**
resetGroup	this attribute determines the name of the group at which the chart expression value is reset.	The value for this attribute must match the name of any group declared in the JRXML report template.
incrementType	This attribute determines when to recalculate the value of the chart expression.	None, Report, Page, Column, Group. Default value is **"None"**.
incrementGroup	This attribute determines the name of the group at which the chart expression is recalculated.	The value for this attribute must match the name of a group declared in the JRXML report template.

Table below summarizes the sub elements of the element <dataset>

Sub element	Description
<incrementWhenExpression>	The way a chart dataset is incremented can be customized by filtering out unwanted data through the use of this sub element.
<datasetRun>	This contains information required to instantiate a report subdataset.

Dataset Types

Specific dataset types are explained below:

Pie Dataset

A pie dataset is characterized by the following expressions:

- <keyExpression> - represent the categories that will make up the slices in the pie chart. This expression can return any java.lang.Comparable object.
- <valueExpression > - produces the values that correspond to each category/key in the dataset. Values are always java.lang.Number objects.
- <labelExpression> - if this expression is missing, the chart will display default labels for each slice in the pie chart. Use this expression, which returns java.lang.String values, to customize the item labels for the pie chart.
- <sectionHyperlink> - sets hyperlinks associated with pie sections.

Category Dataset

A category dataset is characterized by the <categorySeries > element, which contains:

- <seriesExpression > - indicates the name of the series. This expression can return any java.lang.Comparable object.

- <categoryExpression > - returns the name of the category for each value inside the series specified by the series expression. Categories are java.lang.Comparable objects.
- <valueExpression > - produces the values that correspond to each category in the dataset. Values are always java.lang.Number objects.
- <labelExpression > - if this expression is missing, the chart will display default labels for each item in the chart. Use this expression, which returns java.lang.String values, to customize the item labels for the chart.
- <itemHyperlink > - sets hyperlinks associated with chart items

XY Dataset

An XY dataset is characterized by the <xySeries > element, which contains:

- <seriesExpression > - indicates the name of the series. This expression can return any java.lang.Comparable object.
- <xValueExpression > - returns the java.lang.Number value representing the X value from the (x, y) pair that will be added to the current data series.
- <yValueExpression > - returns the java.lang.Number value representing the Y value from the (x, y) pair that will be added to the current data series.
- <labelExpression > - if this expression is missing, the chart will display default labels for each item in the chart. Use this expression, which returns java.lang.String values, to customize the item labels for the chart.
- <itemHyperlink > - sets hyperlinks associated with chart items.

XYZ Dataset

An XYZ dataset is characterized by the <xyzSeries > element, which contains:

- <seriesExpression > - indicates the name of the series. This expression can return any java.lang.Comparable object.
- <xValueExpression > - returns the java.lang.Number value representing the X value from the (x, y, z) item that will be added to the current data series.
- <yValueExpression > - returns the java.lang.Number value representing the Y value from the (x, y, z) item that will be added to the current data series.
- <zValueExpression > - returns the java.lang.Number value representing the Z value from the (x, y, z) item that will be added to the current data series.
- <labelExpression > - if this expression is missing, the chart will display default labels for each item in the chart. Use this expression, which returns java.lang.String values, to customize the item labels for the chart.
- <itemHyperlink > - sets hyperlinks associated with chart items.

Time Series Dataset

A time series dataset is characterized by the timePeriod attribute, and the <timeSeries > element. The timePeriod attribute specifies the type of the data series inside the dataset. Time series can contain numeric values associated with days, months, years, or other predefined time periods. Possible values are: *Year, Quarter, Month, Week, Day - this is the default value, Hour, Minute, Second, Millisecond.*

The <timeSeries > element contains:

- <seriesExpression > - indicates the name of the series. This expression can return any java.lang.Comparable object.
- <timePeriodExpression > - returns a java.util.Date value from which the engine will extract the corresponding time period depending on the value set for the timePeriod attribute mentioned above.
- <valueExpression > - returns the java.lang.Number value to associate with the corresponding time period value when incrementing the current series of the dataset.
- <labelExpression > - if this expression is missing, the chart will display default labels for each item in the chart. Use this expression, which returns java.lang.String values, to customize the item labels for the chart.
- <itemHyperlink > - sets hyperlinks associated with chart items.

Time Period Dataset

A time period dataset is characterized by the <timePeriodSeries > element which contains:

- <seriesExpression > - indicates the name of the series. This expression can return any java.lang.Comparable object.
- <startDateExpression > - specifies the beginning of the date interval with which the numeric value will be associated when it is added to the time period series.
- <endDateExpression > - specifies the end of the date interval with which the numeric value will be associated when it is added to the time period series.
- <valueExpression > - returns the java.lang.Number value to associate with the current date interval specified by the start date and end date expressions.
- <labelExpression > - if this expression is missing, the chart will display default labels for each item in the chart. Use this expression, which returns java.lang.String values, to customize the item labels for the chart.
- <itemHyperlink > - sets hyperlinks associated with chart items.

High Low Dataset

A high low dataset is characterized by the following expressions:

- <seriesExpression > - currently only one series is supported inside a High-Low or Candlestick chart. However, this single series must be identified by a java.lang.Comparable value returned by this expression, and it must also be used as the series name in the chart's legend.
- <dateExpression > - returns the date to which the current (high, low, open, close, volume) item refers.
- <highExpression > - returns a java.lang.Number value, which will be part of the data item added to the series when the dataset gets incremented.
- <lowExpression > - returns a java.lang.Number value, which will be part of the data item added to the series when the dataset gets incremented.
- <openExpression > - returns a java.lang.Number value, which will be part of the data item added to the series when the dataset gets incremented.
- <closeExpression > - returns a java.lang.Number value, which will be part of the data item added to the series when the dataset gets incremented.

- <volumeExpression > - a numeric expression that returns the volume value to use for the current data item. It is used only for Candlestick charts.
- <itemHyperlink > - sets hyperlinks associated with chart items.

Value Dataset

This is a special chart dataset implementation that contains a single value and is used for rendering Meter and Thermometer charts. The value is collected using the <valueExpression > expression.

Chart Plots

Another common JRXML element through all chart types is the **<plot>** element. This allows us to define several of chart's characteristics like orientation and background color. Plots differ based on the type of chart.

Plot Attribute

The table below summarizes the attributes of <plot> element.

Attribute	Description	Values
backcolor	This attribute defines the chart's background color.	Any six digit hexadecimal value is a valid value for this attribute. The hexadecimal value must be preceded by a #.
orientation	This attribute defines the chart's orientation.	Horizontal,Vertical Default value is **"Vertical"**
backgroundAlpha	This attribute defines the transparency of the chart's background color.	The valid values for this attribute include any decimal number between 0 and 1, inclusive. The higher the number, the less transparent the background will be. Default value is **"1"**.
foregroundAlpha	This attribute defines the transparency of the chart's foreground colors.	The valid values for this attribute include any decimal number between 0 and 1, inclusive. The higher the number, the less transparent the background will be. Default value is **"1"**.
labelRotation	This attribute allows rotation of text labels on x-axis to rotate clockwise or anti-clockwise. This attribute applies only to charts for which the x axis is not numeric or does not display dates.	Default value is **"0.0"**.

The <plot> element has a subelement <seriesColor> whose attributes are: *seriesOrder* and *color*. This element customizes colors for series, and their position within in the color sequence.

Specific Settings for Chart Plots

- **piePlot**: It has no specific settings
- **pie3DPlot**: Contains the *depthFactor* attribute, a numeric value ranging from 0 to 1 that represents the depth of

the pie as a percentage of the height of the plot area.

- **barPlot**: One can show or hide tick labels, tick marks or item labels, and provides settings for both axis.
- **bar3DPlot**: provides the same settings as the barPlot, and generates a 3D effect using the xOffset and yOffset attributes.
- **linePlot**: One can show or hide lines connecting item points, can show or hide shapes associated with item points, and provides settings for both axis.
- **scatterPlot**: Similar to the linePlot, it can show or hide lines connecting item points, can show or hide shapes associated with item points, and provides settings for both axis.
- **areaPlot**: Provides settings for both axis.
- **bubblePlot**: One can set the bubble dimensions by setting the scaleType attribute, and provides settings for both axis.
- **timeSeriesPlot**: One can show or hide lines connecting item points, can show or hide shapes associated with item points, and provides settings for both axis.
- **highLowPlot**: One can show or hide open ticks, can show or hide close ticks, and provides settings for both axis.
- **candlestickPlot**: One can show or hide the volume, and provides settings for both axis.
- **meterPlot**: Contains specific settings for the dial shape, scale angle, measurement units, tick interval, dial color, needle color, tick color, value display font, color and format pattern, data range and meter intervals.
- **thermometerPlot**: Contains specific settings for the value location, mercury color, show/hide value lines, value display font, color and format pattern, data range, low range, medium range and high range.
- **multiAxisChart** : Contains specific settings for axis included in the plot

Types of Charts

JasperReports offers built-in support for several chart types. They are listed as below:

- **pieChart**: A combination of a Pie dataset and a Pie plot.
- **pie3DChart**: Groups a Pie dataset and a Pie 3D plot.
- **barChart**: A basic combination of a Category dataset and a Bar plot.
- **bar3DChart**: Wraps a Category dataset and a Bar 3D plot.
- **xyBarChart**: Supports Time Period datasets, Time Series datasets, and XY datasets, and uses a Bar plot to render the axis and the items.
- **stackedBarChart**: Uses data from a Category dataset and renders its content using a Bar plot.
- **stackedBar3DChart**: Uses data from a Category dataset and renders its content using a Bar 3D plot.
- **lineChart**: Groups a Category dataset and a Line plot.
- **xyLineChart**: Groups an XY dataset and a Line plot.
- **areaChart**: Items from a Category dataset are rendered using an Area plot.

- **stackedAreaChart**: Items from a Category dataset are rendered using an Area plot.
- **xyAreaChart**: Uses data from an XY dataset and renders it through an Area plot.
- **scatterChart**: Wraps an XY dataset with a Scatter plot.
- **bubbleChart** : Combines an XYZ dataset with a Bubble plot.
- **timeSeriesChart** : Groups a Time Series dataset and a Time Series plot.
- **highLowChart** : A combination of a High-Low dataset and a High-Low plot.
- **candlestickChart** : Uses data from a High-Low dataset but with a special Candlestick plot.
- **meterChart** : Displays a single value from a Value dataset on a dial, using rendering options from a Meter plot.
- **thermometerChart** : Displays the single value in a Value dataset using rendering options from a Thermometer plot.
- **multiAxisChart** : Contains multiple range axes, all sharing a common domain axis.

Example

To demonstrate the charts, let's write a new report template (jasper_report_template.jrxml). Here we will add the **<barChart>** element to the <pageHeader> section and **<pieChart>** to <summary> section. We would be displaying in charts the marks obtained for each subject. Save it to the directory **C:\tools\jasperreports-5.0.1\test** directory. The contents of the file are as below:

```
<?xml version="1.0" encoding="UTF-8"?>
<jasperReport xmlns="http://jasperreports.sourceforge.net/jasperreports"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://jasperreports.sourceforge.net/jasperreports
http://jasperreports.sourceforge.net/xsd/jasperreport.xsd"
name="jasper_report_template" pageWidth="595" pageHeight="860"
columnWidth="515" leftMargin="40" rightMargin="40"
topMargin="50" bottomMargin="50">

   <field name="subjectName" />
   <field name="marks" />
   <variable name="countNumber"
      calculation="Count">
      <variableExpression>
         <![CDATA[Boolean.TRUE]]>
      </variableExpression>
      </variable>
   <background>
      <band splitType="Stretch"/>
   </background>
   <title>
      <band height="79" splitType="Stretch"/>
   </title>
   <pageHeader>
   <band height="200">
   <barChart>
      <chart evaluationTime="Report">
         <reportElement x="0" y="0" width="555" height="200"/>
         <chartTitle>
            <titleExpression>
               <![CDATA["My First JR Bar Chart"]]>
            </titleExpression>
         </chartTitle>
      </chart>
      <categoryDataset>
         <dataset incrementType="None"/>
```

```
         <categorySeries>
            <seriesExpression>
               <![CDATA[$F{subjectName}]]>
            </seriesExpression>
            <categoryExpression>
               <![CDATA[$F{subjectName}]]>
            </categoryExpression>
            <valueExpression>
               <![CDATA[$F{marks}]]>
            </valueExpression>

         </categorySeries>
      </categoryDataset>
      <barPlot isShowTickMarks="false">
         <plot/>
      </barPlot>
   </barChart>
   </band>

   </pageHeader>
   <columnHeader>
      <band height="20" splitType="Stretch"/>
   </columnHeader>
   <detail>
      <band height="20" splitType="Stretch"/>
   </detail>
   <columnFooter>
      <band height="20" splitType="Stretch"/>
   </columnFooter>
   <pageFooter>
      <band height="20" splitType="Stretch"/>
   </pageFooter>
   <summary>
      <band height="400" splitType="Stretch">
      <pieChart>
         <chart evaluationTime="Report">
            <reportElement x="135" y="0" width="270" height="300"/>
            <chartTitle>
            <titleExpression>
               <![CDATA["My First JR Pie Chart"]]>
            </titleExpression>
            </chartTitle>
            </chart>
         <pieDataset>
            <dataset incrementType="None"/>
            <keyExpression>
               <![CDATA[$F{subjectName}]]>
            </keyExpression>
            <valueExpression>
               <![CDATA[$F{marks}]]>
            </valueExpression>
         </pieDataset>
         <piePlot>
            <plot/>
            <itemLabel/>
         </piePlot>
      </pieChart>
      </band>
   </summary>
</jasperReport>
```

The details of the above file are as below:

- The JRXML element used to create a bar chart is </barChart> in the <pageHeader>. It contains a </chart> sub-element, which contains a <reportElement> sub-element defining the chart's dimensions and position.
- The element in a bar chart must be enclosed between <categoryDataset> and </categoryDataset> JRXML elements.
- <categoryDataset> must contain a <categorySeries> element. This element defines what data element the bars

will represent (subject names, in this example).

- <categoryDataset> must also contain a element, which defines how the data will be separated into categories for comparison. Here data is separated by subject names.
- The <valueExpression> element defines what expression to use to determine the value of each bar in the chart. Here we are using "marks".
- For the pie chart, we have used the element <pieChart> under the <summary> section. It contains a </chart> sub-element.
- The sub-element contains a report expression indicating what to use as a key in the chart. Here we have used subjectName.
- The sub-element contains an expression used to calculate the value for the key. Here we have used marks.

The java codes for report filling remains unchanged. The contents of the file **C:\tools\jasperreports-5.0.1\test\src\com\tutorialspoint\JasperReportFill.java** are as below.

```
package com.tutorialspoint;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.Map;

import net.sf.jasperreports.engine.JRException;
import net.sf.jasperreports.engine.JasperFillManager;
import net.sf.jasperreports.engine.data.JRBeanCollectionDataSource;

public class JasperReportFill {
   @SuppressWarnings("unchecked")
   public static void main(String[] args) {
      String sourceFileName =
      "C://tools/jasperreports-5.0.1/test/jasper_report_template.jasper";

      DataBeanList DataBeanList = new DataBeanList();
      ArrayList<DataBean> dataList = DataBeanList.getDataBeanList();

      JRBeanCollectionDataSource beanColDataSource =
      new JRBeanCollectionDataSource(dataList);

      Map parameters = new HashMap();

      try {
         JasperFillManager.fillReportToFile(
         sourceFileName, parameters, beanColDataSource);
      } catch (JRException e) {
         e.printStackTrace();
      }
   }
}
```

As we would be displaying the marks obtained for each subject, POJO needs to be changed. The file **C:\tools\jasperreports-5.0.1\test\src\com\tutorialspoint\DataBean.java** contents are as below:

```
package com.tutorialspoint;

public class DataBean {
   private String subjectName;
   private Integer marks;

   public String getSubjectName() {
      return subjectName;
   }

   public void setSubjectName(String subjectName) {
```

```
      this.subjectName = subjectName;
   }

   public Integer getMarks() {
      return marks;
   }

   public void setMarks(Integer marks) {
      this.marks = marks;
   }

}
```

Even the contents of the file **C:\tools\jasperreports-5.0.1\test\src\com\tutorialspoint\DataBeanList.java** need to be updated as below:

```
package com.tutorialspoint;

import java.util.ArrayList;

public class DataBeanList {
   public ArrayList getDataBeanList() {
      ArrayList dataBeanList = new ArrayList();

      dataBeanList.add(produce("English", 58));
      dataBeanList.add(produce("SocialStudies", 68));
      dataBeanList.add(produce("Maths", 38));
      dataBeanList.add(produce("Hindi", 88));
      dataBeanList.add(produce("Scince", 78));
      return dataBeanList;
   }

   /*
    * This method returns a DataBean object, with subjectName ,
    * and marks set in it.
    */
   private DataBean produce(String subjectName, Integer marks) {
      DataBean dataBean = new DataBean();

      dataBean.setSubjectName(subjectName);
      dataBean.setMarks(marks);

      return dataBean;
   }
}
```

Report Generation

Next, let's compile and execute the above files using our regular ANT build process. The contents of the file build.xml (saved under directory C:\tools\jasperreports-5.0.1\test) are as below.

> *The import file - baseBuild.xml is picked from chapter [Environment Setup](#) and should be placed in the same directory as the build.xml.*

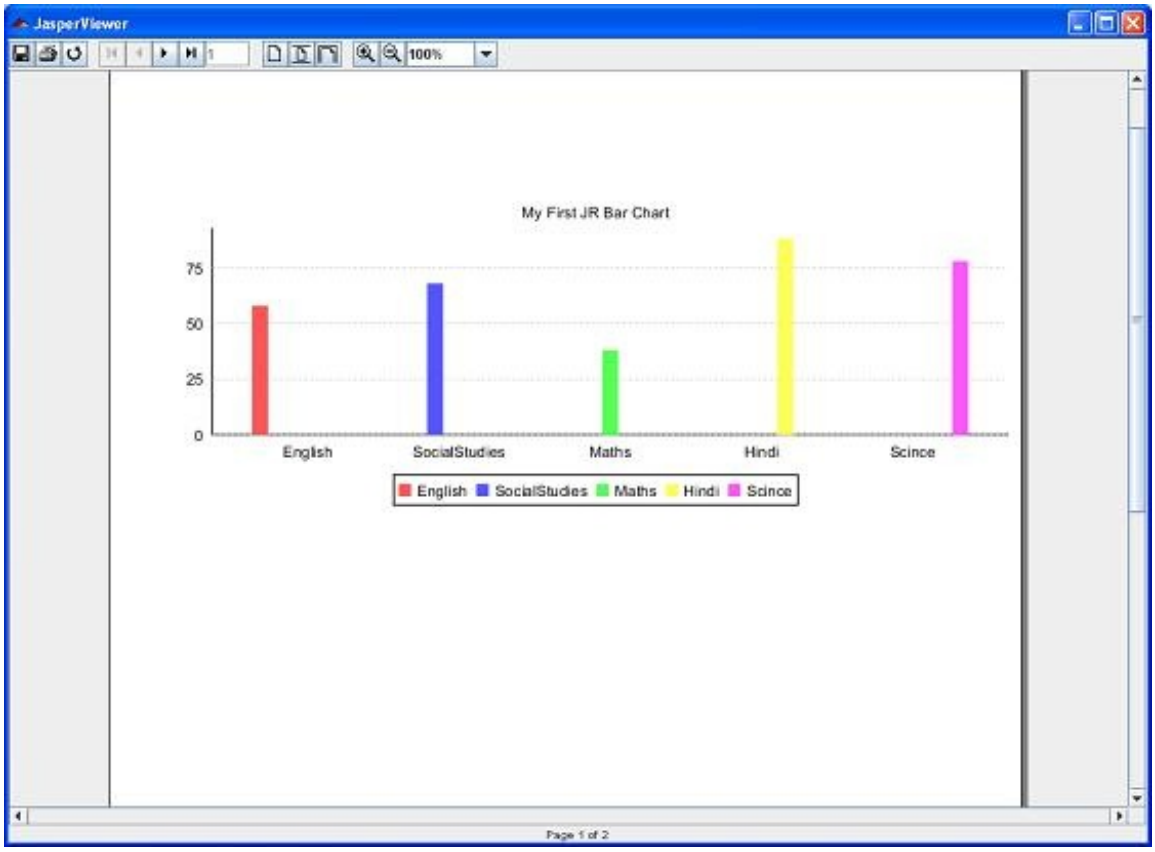
```
<?xml version="1.0" encoding="UTF-8"?>
<project name="JasperReportTest" default="viewFillReport" basedir=".">
   <import file="baseBuild.xml" />
   <target name="viewFillReport"
      depends="compile,compilereportdesing,run"
      description="Launches the report viewer to preview
      the report stored in the .JRprint file.">
      <java classname="net.sf.jasperreports.view.JasperViewer"
      fork="true">
         <arg value="-F${file.name}.JRprint" />
         <classpath ref />
```

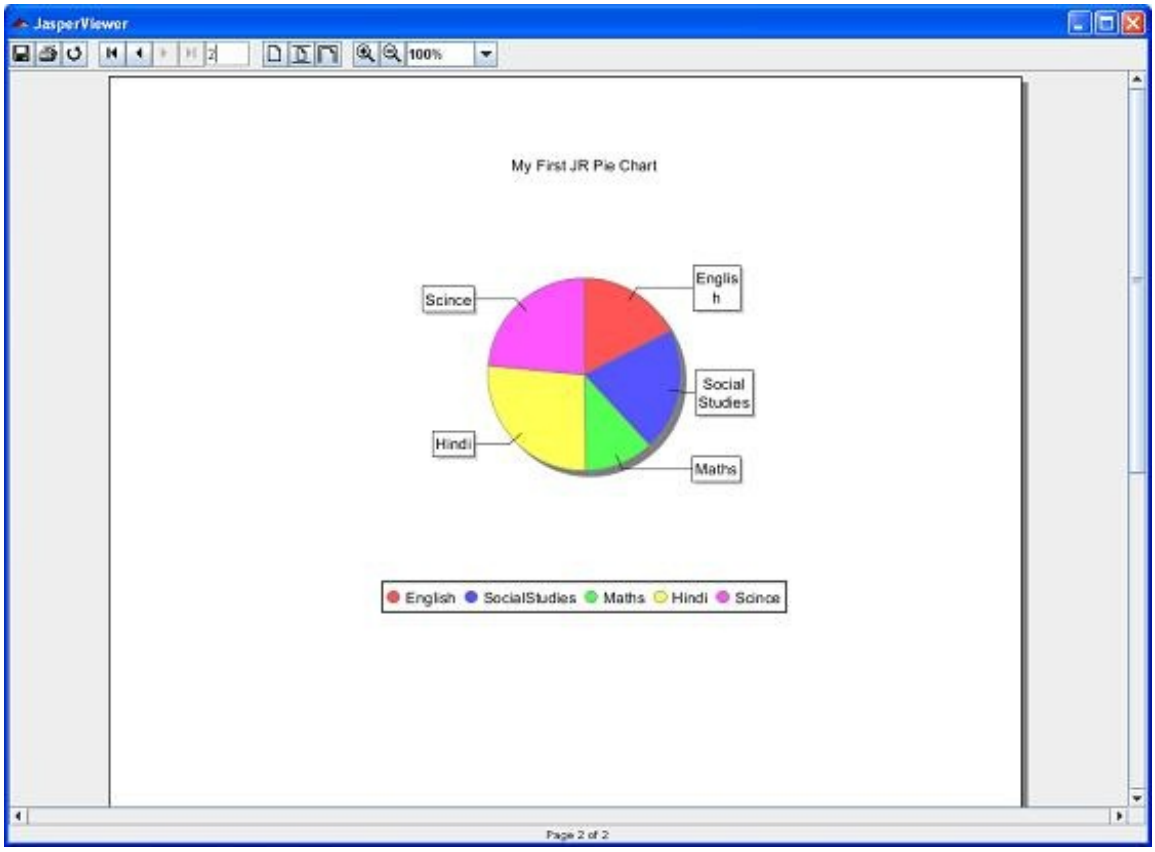
```
      </java>
   </target>
   <target name="compilereportdesing"
      description="Compiles the JXML file and
      produces the .jasper file.">
      <taskdef name="jrc"
      classname="net.sf.jasperreports.ant.JRAntCompileTask">
         <classpath ref />
      </taskdef>
      <jrc destdir=".">
         <src>
         <fileset dir=".">
            <include name="*.jrxml" />
         </fileset>
         </src>
         <classpath ref />
      </jrc>
   </target>
</project>
```

Next, let's open command line window and go to the directory where build.xml is placed. Finally execute the command **ant -Dmain-class=com.tutorialspoint.JasperReportFill** (viewFullReport is the default target) as follows:

```
C:\tools\jasperreports-5.0.1\test>ant -Dmain-class=com.tutorialspoint.JasperReportFill
Buildfile: C:\tools\jasperreports-5.0.1\test\build.xml

clean-sample:
   [delete] Deleting directory C:\tools\jasperreports-5.0.1\test\classes
   [delete] Deleting: C:\tools\jasperreports-5.0.1\test\jasper_report_template.jasper
   [delete] Deleting: C:\tools\jasperreports-5.0.1\test\jasper_report_template.jrprint

compile:
    [mkdir] Created dir: C:\tools\jasperreports-5.0.1\test\classes
    [javac] C:\tools\jasperreports-5.0.1\test\baseBuild.xml:28:
    warning: 'includeantruntime' was not set, defaulting to bu
    [javac] Compiling 3 source files to C:\tools\jasperreports-5.0.1\test\classes

compilereportdesing:
      [jrc] Compiling 1 report design files.
      [jrc] log4j:WARN No appenders could be found for logger
      (net.sf.jasperreports.engine.xml.JRXmlDigesterFactory).
      [jrc] log4j:WARN Please initialize the log4j system properly.
      [jrc] log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
      [jrc] File : C:\tools\jasperreports-5.0.1\test\jasper_report_template.jrxml ... OK.

run:
     [echo] Runnin class : com.tutorialspoint.JasperReportFill
     [java] log4j:WARN No appenders could be found for logger
     (net.sf.jasperreports.extensions.ExtensionsEnvironment).
     [java] log4j:WARN Please initialize the log4j system properly.

viewFillReport:
     [java] log4j:WARN No appenders could be found for logger
     (net.sf.jasperreports.extensions.ExtensionsEnvironment).
     [java] log4j:WARN Please initialize the log4j system properly.

BUILD SUCCESSFUL
Total time: 19 minutes 45 seconds
```

As a result of above compilation, a JasperViewer window opens up as in the screen below:





Here we see that the bar chart is created in the pageheader and the pie chart is created in the summary sections.